UNIVERSIDADE da MADEIRA

CTeSP CURSOS TÉCNICOS SUPERIORES PROFISSIONAIS

Tecnologias e Programação de Sistemas de Informação

Classes e instâncias, cadeias de caracteres

Arquitetura de Dispositivos | David Jardim

Cofinanciado por:

MADEIRA 14-20

REGIÃO AUTÓNOMA DA MADEIRA

PORTUGAL 2020

UNIÃO EUROPEIA
Fundo Social Europeu

Da aula anterior…

- Manipulação de caracteres

- Vetores-multidimensionais

Classes e instâncias

- Classe

Não classe pacote!

  - Modelo para construção de instâncias que partilham conjunto de características observáveis
    - Propriedades
    - Operações

- Instância (ou objecto)
  - Exemplar de classe
  - Construído e manipulado durante execução do programa
  - Tem identidade e estado próprios

Classes (não pacote)

- Nome (singular)
  - Reflecte o que instâncias da classe representam (e.g., `Point`, `Person`, `Game`, `Board`, `Player`)
  - Maiúscula inicial (convenção Java)

- Composição básica em Java
  - Atributos
  - Construtores
  - Métodos de instância (e não de classe)

Métodos de classe são `static`.

Classes: membros

Atributos	Variáveis que cada instância da classe possui em exclusivo. Conjunto de valores dos atributos de uma instância é o seu estado.
Construtores	Rotinas cujo objectivo é construir instâncias da classe, colocando-as num estado válido.
Métodos de instância	Métodos que se invocam através de e para uma instância particular da respectiva classe. Implementam operações usadas (a) para aceder a propriedades de uma instância, (b) para realizar acções sobre essa instância, (c) para realizar acções sobre terceiros, etc.

Classes: atributos e propriedades

- Atributos
  - Variáveis possuídas em exclusivo por cada instância de classe
  - Conjunto dos seus valores determinam estado de uma instância
  - Parte da implementação de uma classe

- Propriedades
  - Acessíveis (em Java) através de operações da classe
  - Conjunto dos seus valores determinam estado *observável* de uma instância
  - Parte da interface de uma classe

- Exemplos
  - `x` e `y` como propriedades (e atributos) de `Point`
  - `cardinality` como propriedade (e atributo) de `IntegerSet`
  - `name` como propriedade (e atributo) de `Person`

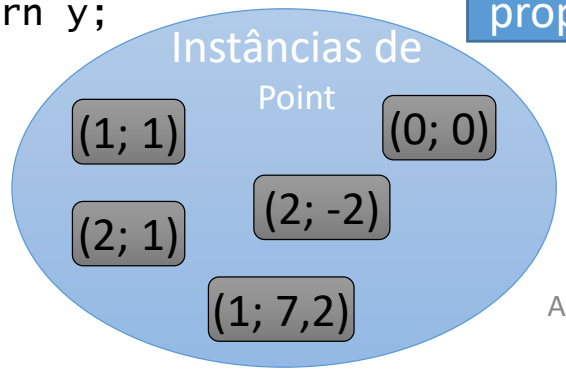
Classes: atributos e propriedades

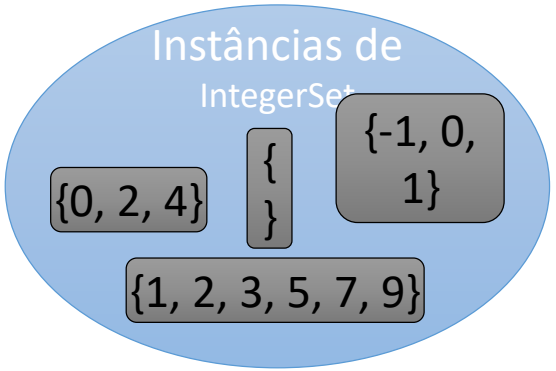
```
public class Point {
    private double x;
    private double y;
    …
    public double getX() {
        return x;
    }

    public double getY() {
        return y;
    }
    …
}
```

atributos

propriedades

```
public class IntegerSet {
    private int[] items;
    private int cardinality;
    …
    public int[] getItems() {
        return
            Arrays.copyOf(items,
                          cardinality);
    }
    …
}
```





Classes: construtores

- Rotinas que constroem instância de uma classe
  - Inicializam atributos
  - Garantem estado inicial válido
  - Uma classe pode ter vários construtores, mas com diferentes assinaturas

```
public class Point {
   private double x;
   private double y;

   public Point() {
      x = 0.0;
      y = 0.0;
   }

   public Point(final double x,
                final double y) {
      this.x = x;
      this.y = y;
   }

   …
}
```

`this`: usado para desambiguar quando variáveis locais (e.g., parâmetros) tiverem mesmo nome que atributos.

Classes: instanciação

- Operador **new**
  - Constrói novas instâncias
  - Invoca construtor da classe

- Exemplos
  - `new Point()`
  - `new Point(1.0, -2.0)`

```
public Point() {
   x = 0.0;
   y = 0.0;
}
```

```
public Point(final double x,
             final double y) {
   this.x = x;
   this.y = y;
}
```

⟸ 1,0
⟸ -2,0

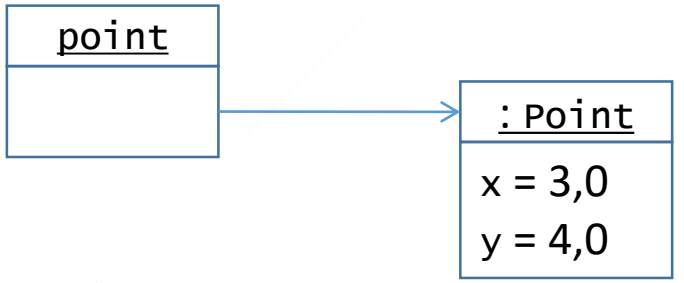
Classes: referências

- Referências são variáveis
- Têm tipo associado
- Referem ou apontam instância desse tipo
- Podem não referenciar nada: valor `null`
- Exemplo
  - `Point point = new Point(3.0, 4.0);`

inicialização da referência

definição da referência

construção da instância



Classes são tipos de referência

- Tipos primitivos (`int`, `double`, etc.)
  - Variáveis guardam valor desse tipo
  - Atribuição altera valor guardado
  - São tipos de valor

- Classes Java
  - Variáveis guardam referência para instância dessa classe (ou classe compatível)
  - Atribuição altera referência guardada
  - Atribuição *não* altera a instância referenciada
  - São tipos de referência

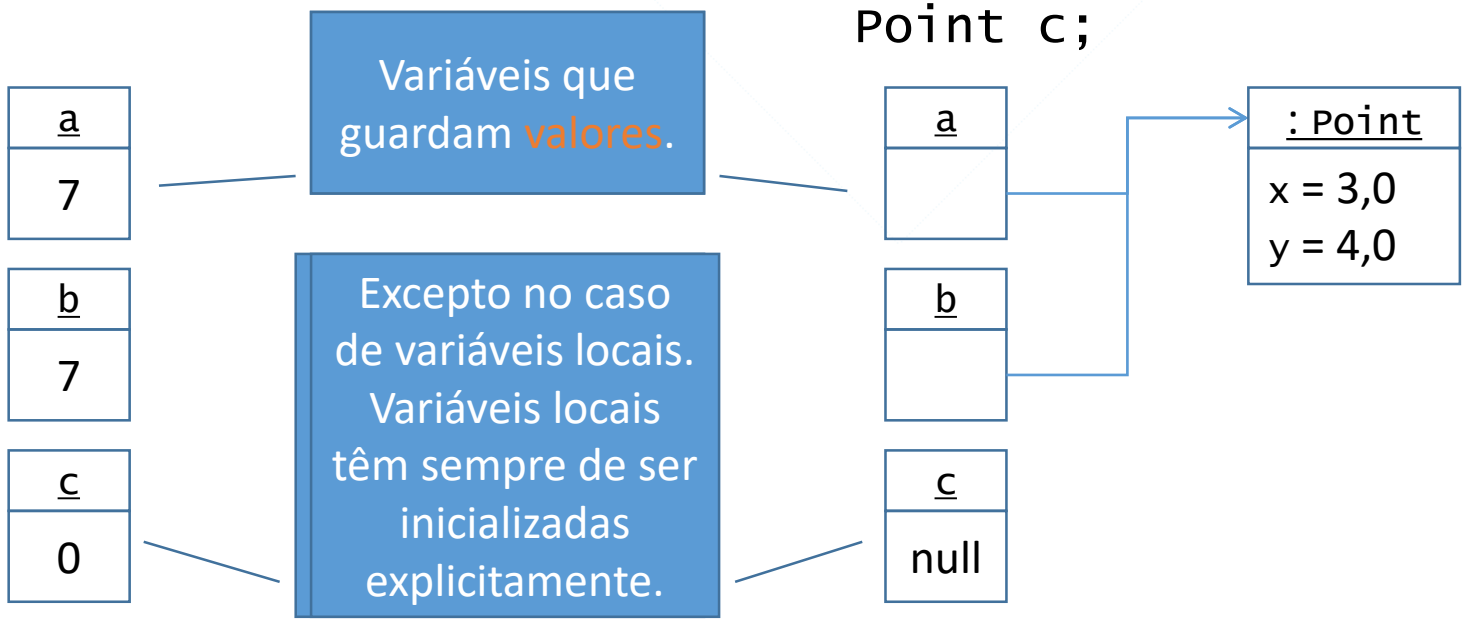
Classes são tipos de referência

Tipos primitivos (`int`, `boolean`, etc.)

```
int a = 7;
int b = a;
int c;
```

Tipos de referência (matrizes, classes)

```
Point a = new Point(3.0, 4.0);
Point b = a;
Point c;
```



Igualdade vs. identidade

Tipos primitivos (`int`, `boolean`, etc.)

```
int a = 7;
int b = a;
```

a
7

b
7

a == b? Sim!

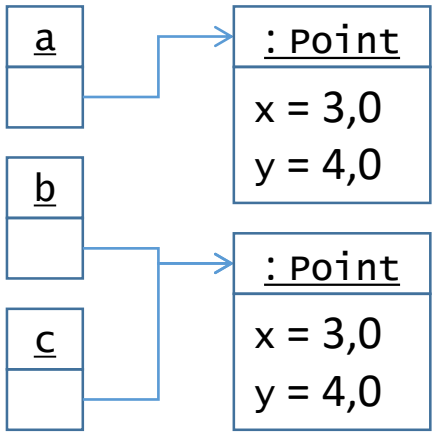
Operador == verifica se valores são iguais!

Operador == verifica se *referências* são iguais! Ou seja, verifica se se referem à mesma instância!

a == b? Não!
c == b? Sim!

Tipos de referência (matrizes, classes)

```
Point a = new Point(3.0, 4.0);
Point b = new Point(3.0, 4.0);
Point c = b;
```



Inicializações automáticas

- Atributos de classes e itens de matrizes de tipos primitivos inicializados com valor por omissão:
  - `int` – `0`
  - `double` – `0.0`
  - `boolean` – `false`
  - …
- Atributos de classes e itens de matrizes de tipos de referência inicializados com `null`

Classes: operações e métodos de instância

- Operações realizam-se sobre uma instância da classe
- Métodos são implementação de operações
- Operações podem ser
  - Funções – Calculam e devolvem um resultado
  - Procedimentos – Realizam uma acção

Classes: funções e procedimentos de instância

- Funções
  - Conjunto de instruções, com interface bem definida, que efectua um dado cálculo
  - Devolvem explicitamente um resultado ao exterior
  - Não devem efectuar qualquer alteração ao estado do objecto

- Procedimentos
  - Conjunto de instruções, com interface bem definida, que realiza uma determinada acção (normalmente alteram o estado da instância)
  - Não devolvem explicitamente um resultado ao exterior

Classes: natureza das operações de instância

- Inspectores (funções) – Acedem às propriedades da instância
- Modificadores (procedimentos) – Agem sobre a instância, modificando-a
- Funções não inspectoras – Acedem às propriedades de terceiros
- Procedimentos não modificadores – Agem sobre terceiros

Classes: funções

```
public class Nome {
    private tipo atributo;

    …

    public tipo nome(parâmetros) {
        instruções
        …
        return expressão;
    }

    …
}
```

os atributos não devem ser modificados pela função

interface

implementação

Classes: procedimentos

```
public class Nome {
    private tipo atributo;

    …

    public void nome(parâmetros) {
        instruções
        …
        return expressão;
    }

    …
}
```

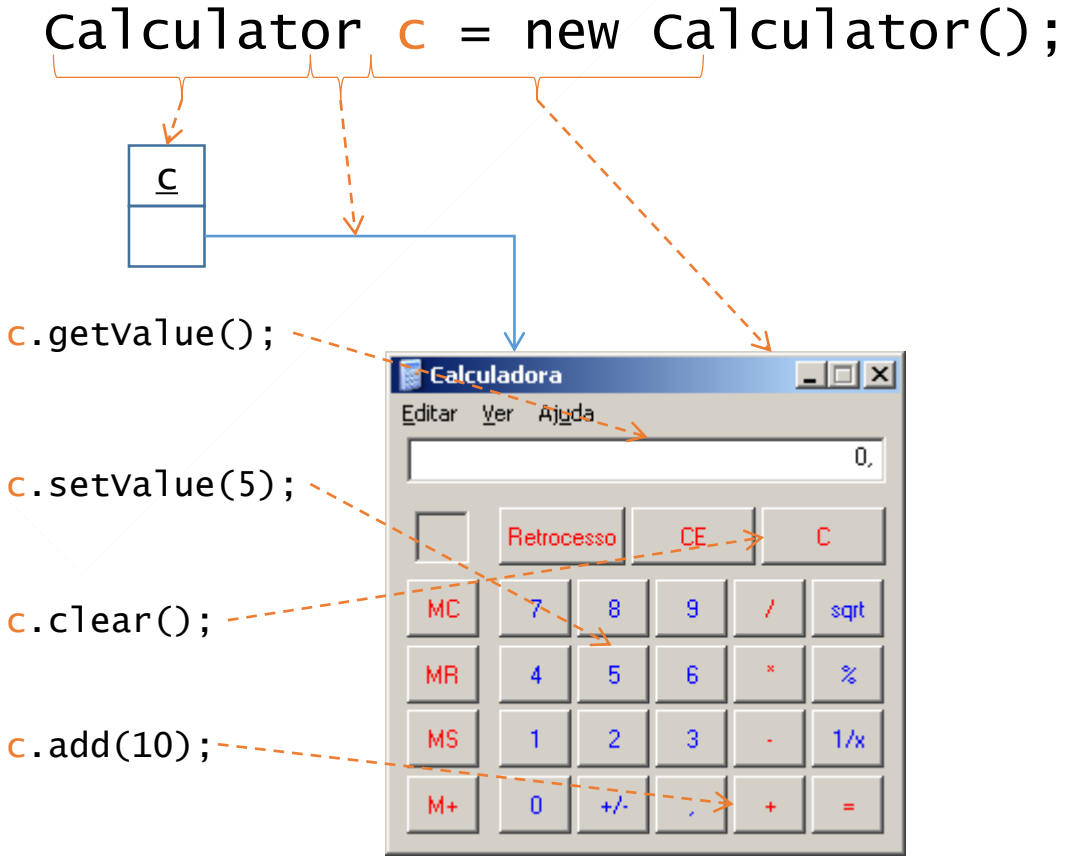
os atributos podem ser modificados pelo procedimento

interface

implementação

Exemplo: calculadora

```
public class Calculator {
   private double value;

   public Calculator() {
      value = 0.0;
   }
   public double getValue() {
      return value;
   }
   public void setValue(
                  double newValue) {
      value = newValue;
   }
   public void clear() {
      value = 0.0;
   }
   public void add(double term) {
      value += term;
   }
}
```



Classe vs. classe pacote

Classe

- Molde para construção de instâncias

```
public class Calculator {

    private int value;

    public Calculator() {…}

    public int value() {…}

    public void clear() {…}
}
```

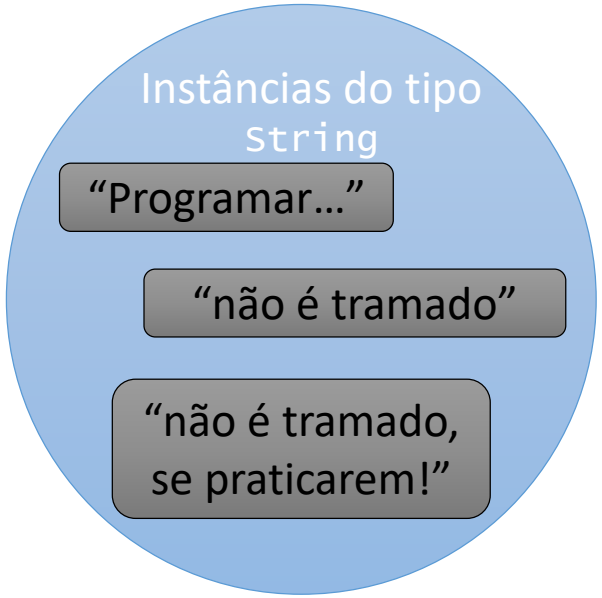
Classe pacote

- Conjunto de métodos de classe relacionados empacotados num módulo

```
public class Math {

    private Math() {…}

    public static
    double sin(double angle) {…}

    public static
    double sqrt(double value) {…}
}
```

Classe String

- Representa cadeias de caracteres
- Instâncias imutáveis, i.e., estado observável não muda após construção
- Valores literais
  - Caracteres entre aspas
  - São referências para instâncias



String: inicialização

- Utilizar cadeias de caracteres literais optimiza memória e aumenta eficiência

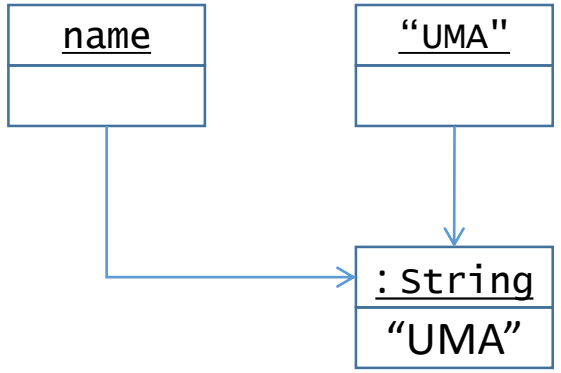
- Evitar utilizar construir novas instâncias

- Exemplo a evitar

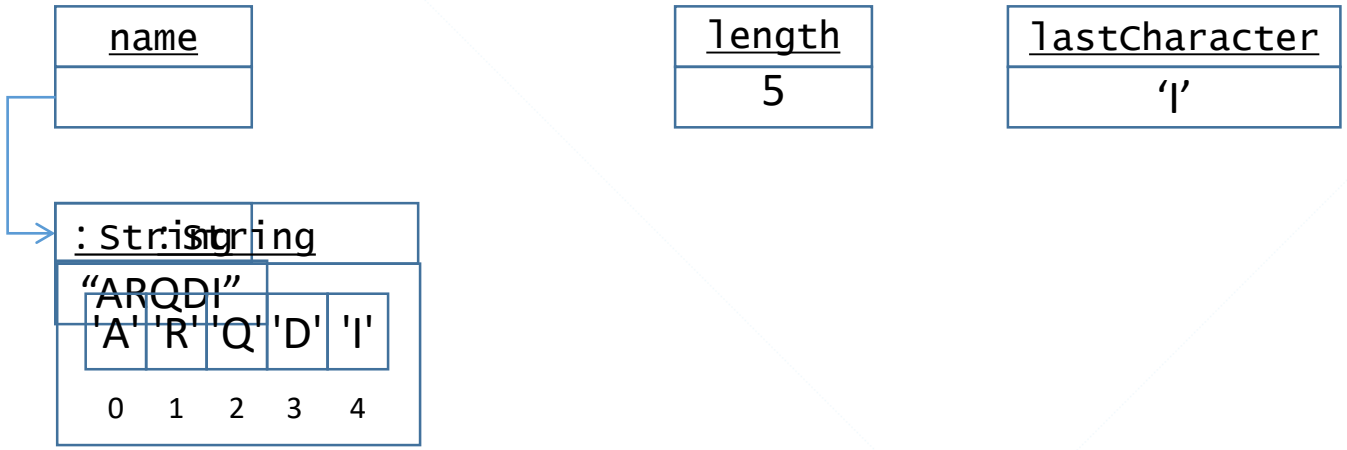
```
String name =
    new String("UMA");
```

```
String name = "UMA";
```

valor literal (referência para instância de `String` pré-existente)



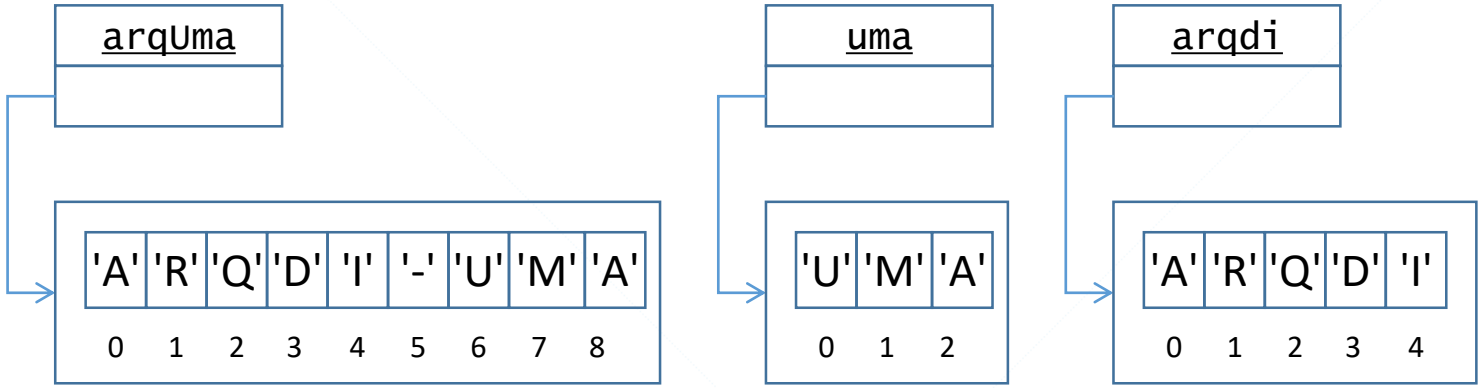
String: comprimento e caracteres



- Comprimento
  - `int length = name.length();`

- Caractere em determinada posição
  - `char lastCharacter = s.charAt(4);`

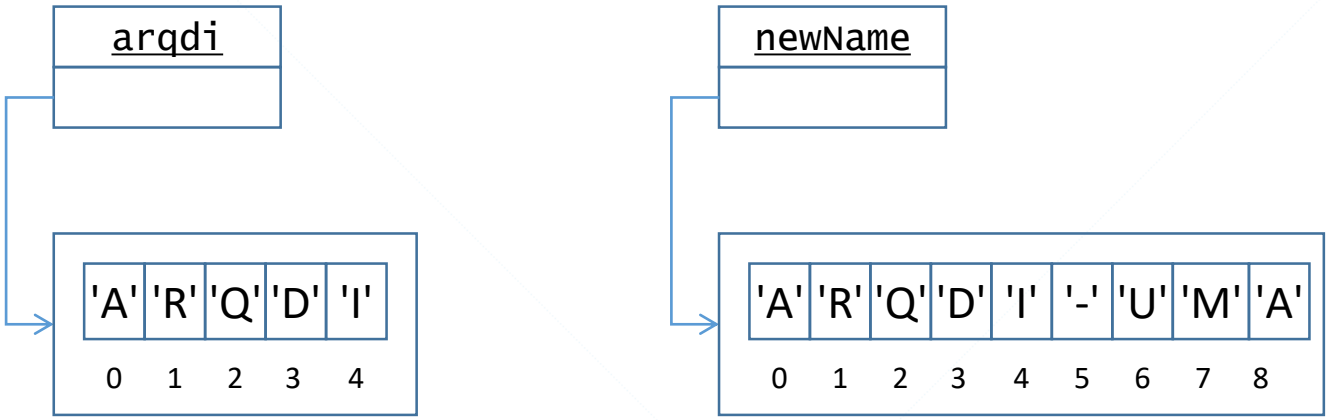
String: subcadeias



- Inicialização
  - `String arqUma = “ARQDI-UMA";`

- Subcadeia
  - `String uma = s.substring(6);`
  - `String arqdi = s.substring(0, 5);`

String: concatenação



- Concatenação
  - `String newName = arqdi.concat("-UMA");`

- Concatenação simplificada (operador +)
  - `String newName = arqdi + "-UMA";`

String: Igualdade vs. identidade

Tipos primitivos (`int`, `boolean`, etc.)

```
int a = 7;
int b = a;
```

Operador == verifica se valores são iguais!

a == b? Sim!

a
7

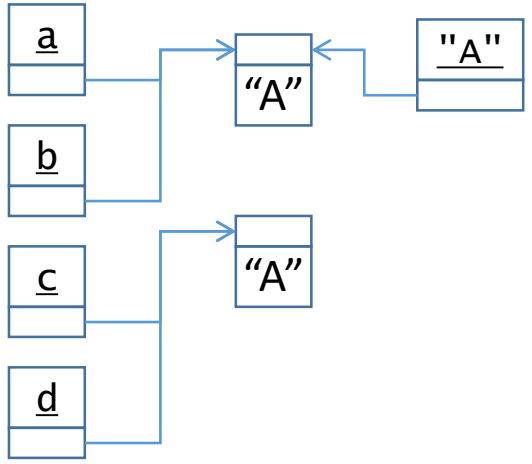
b
7

Operador == verifica se *referências* são iguais! Ou seja, verifica se se referem à mesma instância!

Tipos de referência (matrizes, classes)

```
String a = "A";
String b = "A";
String c = new String("A");
String d = c;
```

a == b? *Sim*!
a == c? Não!
c == d? Sim.
a.equals(b)? Sim.
a.equals(c)? Sim!
c.equals(d)? Sim.



A reter

- Classes e instâncias
  - Instanciação
  - Membros
    - Propriedades vs. atributos
    - Operações vs. métodos
    - Construtores
  - Funções vs. procedimentos
  - Tipos de referência vs. tipos de valor
  - Tipos primitivos vs. restante tipos (em Java)
  - Igualdade vs. identidade

- Cadeias de caracteres com a classe `String`

Sumário

- Classes e instâncias

- Cadeias de caracteres


UNIVERSIDADE da MADEIRA
CTeSP
CURSOS TÉCNICOS
SUPERIORES PROFISSIONAIS